Programmation Evenementielle

Les exemples de code que vous trouverez dans cette leçon ne peuvent malheureusement pas être exécutés dans le contexte de ce notebook à partir du serveur web de cours, car celui-ci n'a pas la possibilité/permission de créer une fenêtre graphique sur votre ordinateur, à l'extérieur de votre fureteur web. Pour que ces exemples puissent fonctionner, il vous faudra télécharger cette leçon sur votre ordinateur, sous la forme d'un fichier avec l'extension .ipynb, et la visualiser en faisant appel au logiciel Jupyter exécuté directement sur votre ordinateur. Ce logiciel est un programme python qui s'installe facilement avec la commande pip. Il suffit d'exécuter en ligne de commande pip3 install jupyter et jupyter sera automatiquement installé sur votre machine. Pour de plus amples détails, parcourir la documentation de Jupyer.

# Procédural vs Événementiel

Jusqu'à présent, nous avons traité des programmes qui séquentiellement :

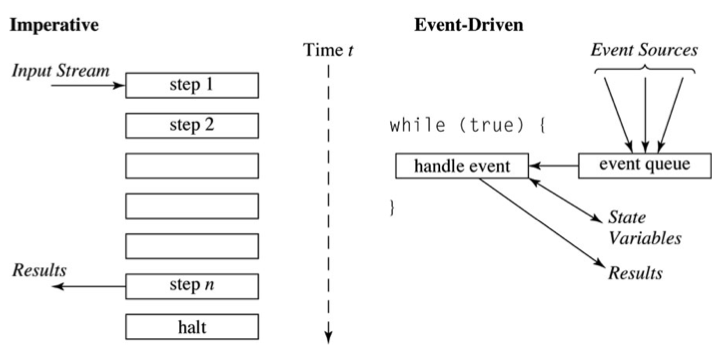
* lisent ou produisent des entrées ;
* effectuent des traitements ;

Et produisent des sorties.

En programmation événementielle, on répond à des événements :

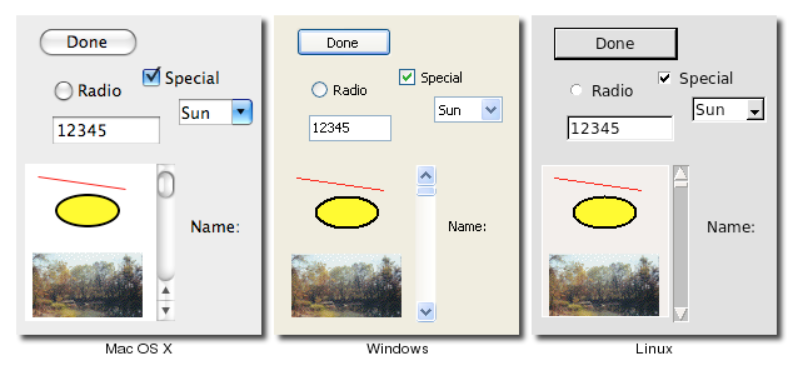
* clics et déplacements de souris ;
* touches de clavier ;
* redimensionnement de fenêtre ;
* etc.

Le déroulement du programme est alors entièrement déterminé par la séquence des événements.



# Widgets

Néologisme qui provient de l'expression « window gadgets ».



Sur l'illustration précédente, on peut voir de haut en bas et de gauche à droite :

* un « button » ;
* un « checkbox » ;
* un « radiobutton » ;
* un « Combobox » ;
* un « Entry » ;
* un « Canvas » ;
* un « Scrollbar » ;
* un « Label».

# Événements

L'utilisateur interagit avec un widget :

* par exemple, il clique sur un bouton... ;
* le bouton appelle une fonction spécifiée par le programmeur ;
* cette fonction s'appelle un « callback ».

# Traitement des événements

En programmation événementielle, le traitement des événements se fait habituellement de façon automatique :

* le système accumule les événements dans une file d'attente ;
* les événements sont transmis aux widgets ;
* chaque widget appelle la fonction (« callback ») définie par le programmeur.

# Gestion des Widgets

Une interface est définie par une hiérarchie de widgets à l'intérieur d'une fenêtre.

Des mécanismes permettent de positionner les widgets les uns par rapport aux autres :

* les widgets ont habituellement une dimension « naturelle » ;
* ces mécanismes permettent de spécifier l'agencement des widgets dans la fenêtre ;
* et comment cet agencement doit changer lorsqu'on change la dimension de la fenêtre.

# Module tkinter

Ce module est une interface Python sur la librairie Tcl/Tk :

* tkinter signifie « Tk interface » ;
* Tcl signifie « Tool command language » (se prononce « tickle ») ;
* Tk veut dire « Toolkit » (boite à outils).

# Widget Label

Voici les principaux attributs du widget Label :

* anchor = NW|N|NE|W|E|SW|S|SE
  + point d'ancrage de la chaîne.
* background = couleur
  + couleur de l'arrière plan.
* foreground = couleur
  + couleur du texte.
* height = n
  + hauteur du widget en nombre de lignes.
* justify = LEFT|CENTER|RIGHT
  + justification dans le cas d'une chaîne de plusieurs lignes.
* padx = n
  + espacement horizontal en pixels.
* pady = n
  + espacement vertical en pixels
* text = chaîne
  + chaîne de caractères du widget.
* width = n
  + largeur du widget en nombre de caractères

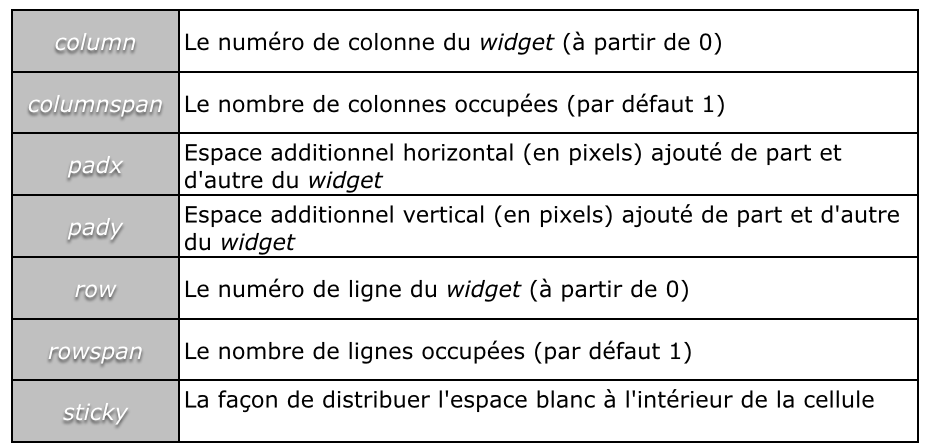
# Attribut des couleurs

* background = couleur
  + couleur de l'arrière plan.
* foreground = couleur
  + couleur du texte.
* height = n
  + hauteur du widget en nombre de lignes.
* justify = LEFT|CENTER|RIGHT
  + justification dans le cas d'une chaîne de plusieurs lignes.
* padx = n
  + espacement horizontal en pixels.
* pady = n
  + espacement vertical en pixels
* text = chaîne
  + chaîne de caractères du widget.
* width = n
  + largeur du widget en nombre de caractères

# Positionnement des widgets

Par défaut les widgets sont centrés en une seule colonne verticale, mais ce comportement peut-être changé avec les arguments suivants de la méthode grid :

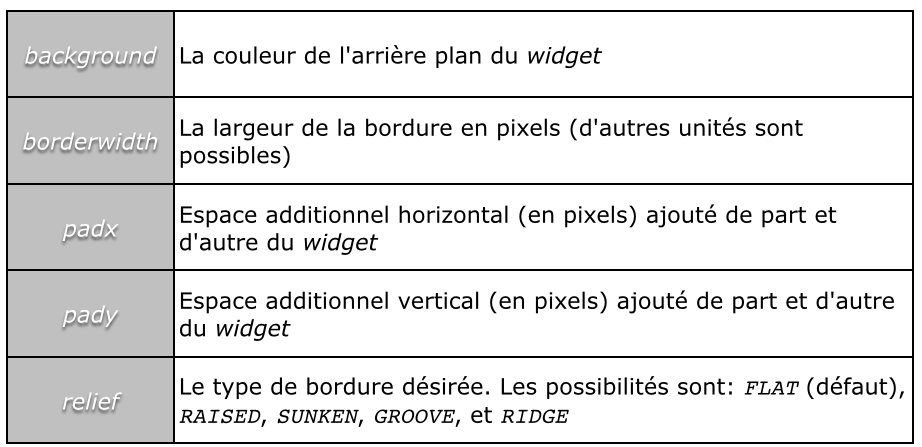
* widget.grid(option = value)



# Widget Frame

Permet de créer un cadre pour contenir des widgets :

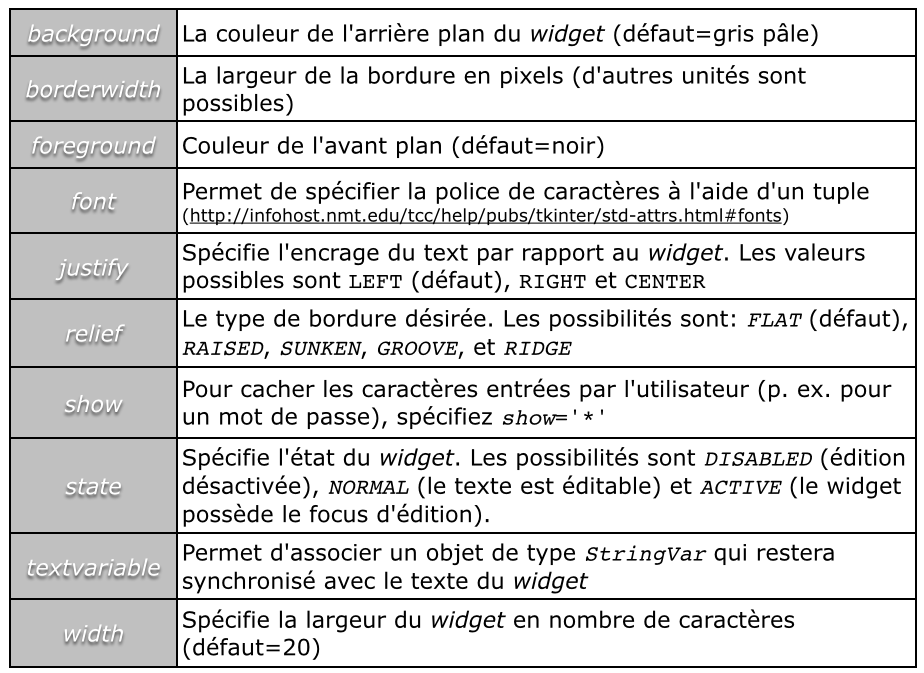
w = tk.Frame(parent, option=valeur, ...)



# Widget Entry

Permet à l'utilisateur d'éditer une ligne de texte dans un rectangle :

w = Entry(parent, option=valeur, ...)



Un autre widget nommé Text permet d'éditer un texte de plusieurs lignes.

# Méthode Bind

Disponible pour tous les widgets.

Permet d'attacher des événements à des widgets à travers une fonction :

* p.ex. une touche de clavier spécifique à une boite d'entrée («Entry»)
* lorsque le widget aura le «focus» du clavier et que la touche en question sera pressée par l'utilisateur, alors tkinter appelera la fonction spécifiée

champ = Entry(parent)

champ.grid()

champ.bind('<KeyPress-Return>', fonction)

Pour de plus amples détails : <http://infohost.nmt.edu/tcc/help/pubs/tkinter/events.html>

# Traitement des événements

Deux formes :

* def handlerName(event):
* def handlerName(self, event):

L'argument event est un objet de type Event qui contient toute l'information relative à l'événement.

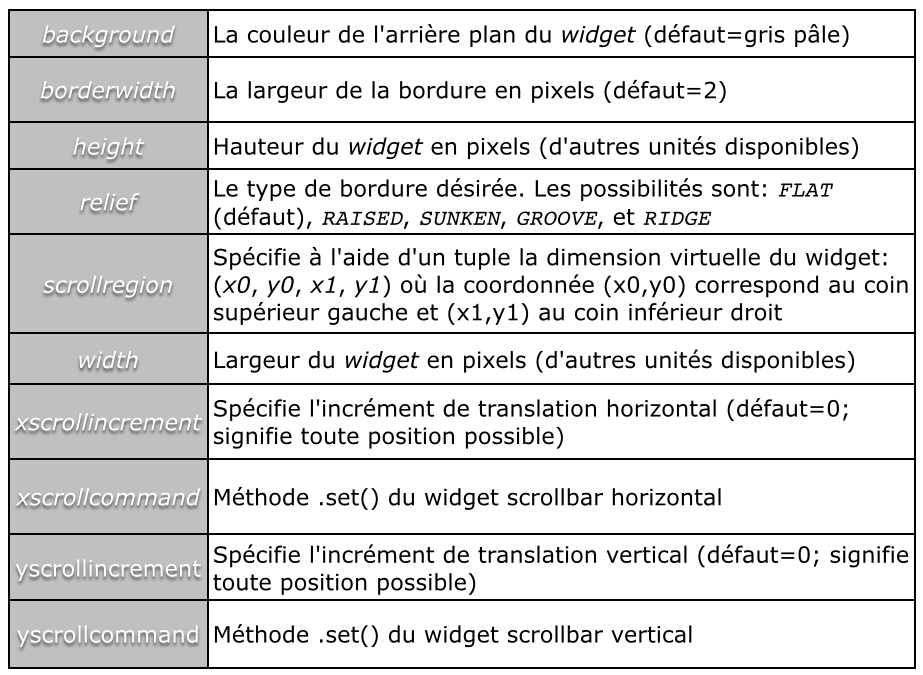
Pour de plus amples détails : <http://infohost.nmt.edu/tcc/help/pubs/tkinter/events.html#event-handlers>

# Widget Canvas

Un canvas est une région rectangulaire dans laquelle on peut dessiner avec différentes primitives graphiques :

* arcs de cercle et ellipses ;
* bitmap et images ;
* lignes, rectangles et polygones ;
* text.

Les principaux attributs du canvas sont :



Tous les détails se retrouvent à l'adresse :

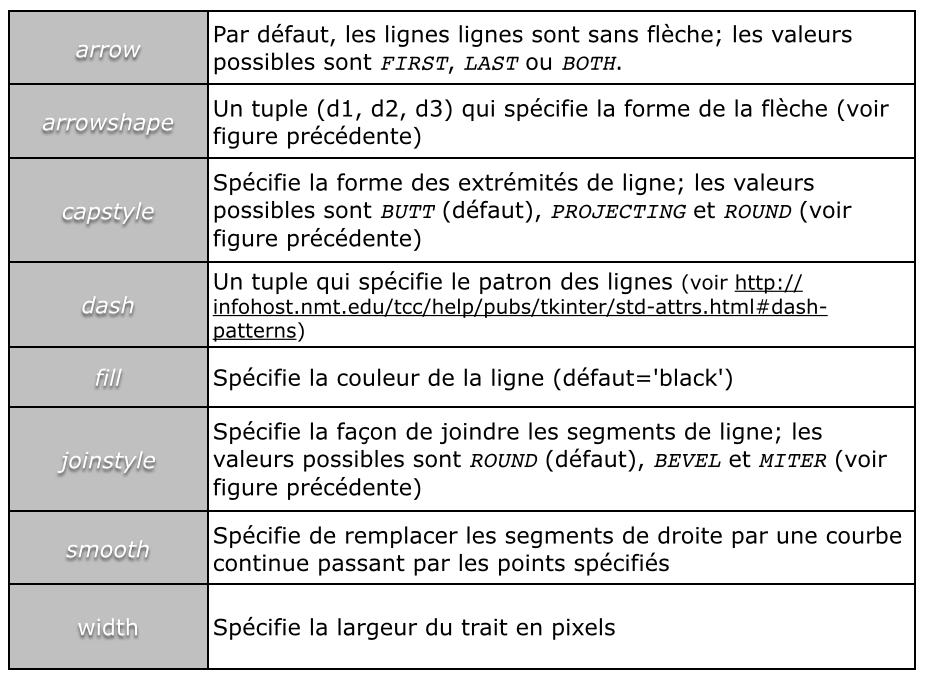
<http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/canvas.html>

L’origine d’un canevas est en haut à gauche.

# Lignes

On peut tracer une ligne de la façon suivante :

* canvas = Canvas(parent)
* id = canvas.create\_line(x0, y0, x1, y1, ..., option=valeur, ...)



Pour de plus amples détails :

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/create_line.html>
* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/std-attrs.html#cap-join-styles>

# Autres liens

**create\_arc**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/create_arc.html>

**create\_bitmap**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/create_bitmap.html>

**create\_image**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/create_image.html>

**LabelFrame**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/labelframe.html>.

**ListBox**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/listbox.html>.

**Menu**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/menu.html>.

**MenuButton**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/menubutton.html>.

**Scale**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/scale.html>.

**ScrollBar**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/scrollbar.html>.

**SpinBox**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/spinbox.html>.

**messagebox**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/tkMessageBox.html>

**filedialog**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/tkFileDialog.html>

**colorchooser**

* <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/tkColorChooser.html>